TerraFusion metadata generation documentation

Author: Yat Long Lo (Richie)

Email: yllo2@illinois.edu

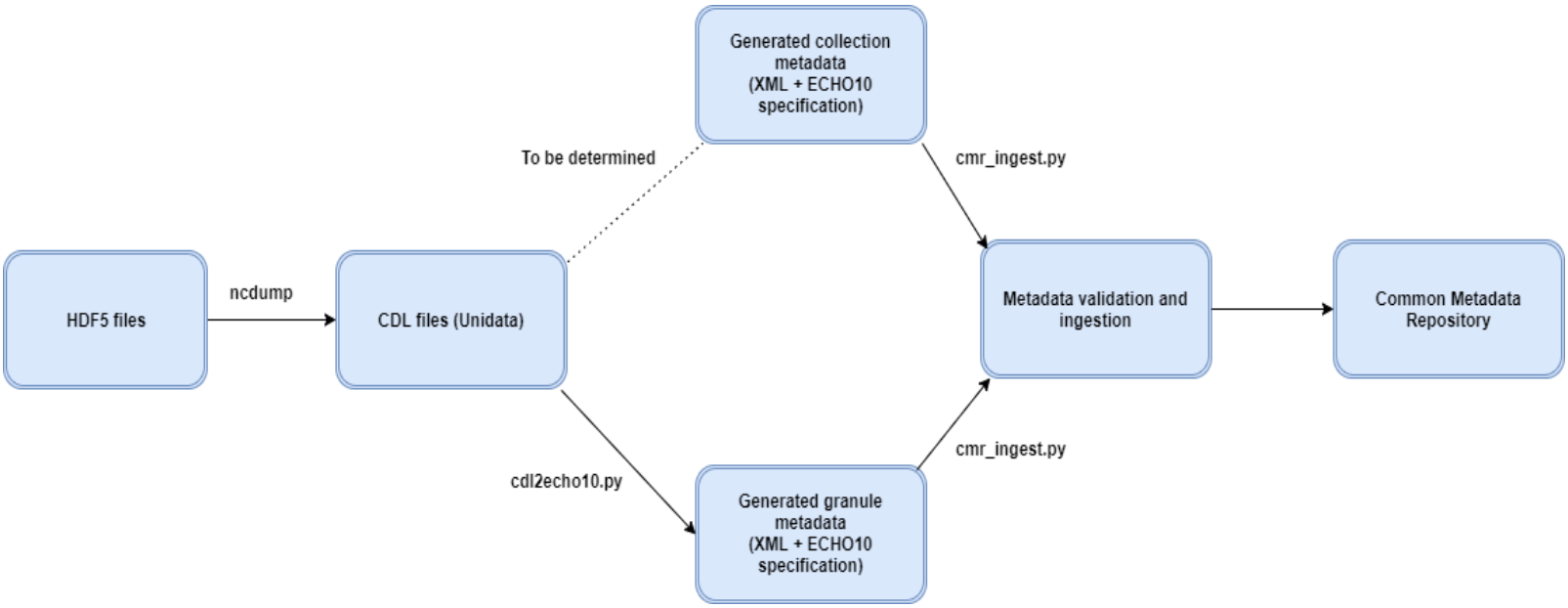
Repository: https://github.com/TerraFusion/basicFusion/tree/master/CMR

1. Purpose

This document aims to provide detailed guide for future users of the TerraFusion metadata generation program. The codes provide functionalities to generate corresponding metadata files for successful ingestion into NASA's Common Metadata Repository (CMR)*.

*Collection metadata generation is still in development. Implementation details are yet to be confirmed.

2. General flow



The figure above describes the general flow of the process from XML generation to ingestion stage. Three main actions/programs have to be used to complete the whole flow successfully, namely ncdump, cdl2echo10.py and cmr_ingest.py. Primary purpose of each of them are given below. Detailed descriptions and usages of cdl2echo10.py and cmr_ingest.py will be given in later sections.

ncdump: generates CDL files (Needed for metadata generation) from HDF5 files
cdl2echo10.py: generates metadata XML files from CDL input files in NASA's ECHO10 specification

cmr_ingest_py: performs metadata validation and ingestion with NASA's CMR server*

*Proper credentials are needed

3. cdl2echo10.py

Use: It generates XML metadata in the ECHO10 specification, from the CDL file input.

Input parameters: CDL_filename, destination_XML_filename, doctype(-c/-g) *

*-c – collection, -g – granule

Methodology: The program extracts InputGranules names from the CDL file, which are used to query the CMR's server for metadata. The data extracted are entirely based on ECHO10's specification*. After data retrieval, the whole XML ElementTree would be written to the XML filename specified.

* https://wiki.earthdata.nasa.gov/display/CMR/CMR+Client+Partner+User+Guide

Python Libraries required: sys, os, logging, ply, netCDF4, lxml, numpy urllib2, re, datetime

4. cmr_ingest.py

Use: It takes any generated metadata XML files (from cdl2echo10.py) and ingest them into NASA's CMR or you can update those that have been ingested. A CMR account is required for the process, contact relevant parties like CMR support for authorization. A credential XML file has to be passed in, containing the credentials of your account, a sample can be found on the repository – mytokengenerator.xml.

Input parameters: metadata_filename, credential_file, doctype(-c/-g) *, action(-u, -i)*

*-c – collection, -g – granule

*-u – update, -i – ingest

Methodology: The program would take in the credential file and generate an API token by calling CMR's API. Based on your input parameters, the program would first validate your metadata file submitting to CMR's server. If validated, the program would proceed to update or ingest your metadata.

Python Libraries required: sys, requests, json, xml

5. Next step

- Details of collection generation needs to be confirmed and implemented. At the moment, an example collection metadata file has been compiled on the repo (sample_echo10.xml), which has been successfully validated and ingested. Future development can take reference from that.

- Depending on the group's preference, it may be a good idea to put generation and ingestion into one single script. As the only dependence between the 2 is the generated XML file, the combination should be fairly straightforward.